UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Karmen Knavs

Mobilna aplikacija za učenje črk

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2015

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Karmen Knavs

Mobilna aplikacija za učenje črk

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Alenka Kavčič

Ljubljana, 2015

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Mobilna aplikacija za učenje črk

Tematika naloge:

V okviru diplomske naloge izdelajte mobilno aplikacijo za predšolske in šolske otroke, ki otroku pomaga pri spoznavanju abecede, posameznih črk in pri učenju pisanja črk. Aplikacija za učenje črk naj bo zasnovana na igri, tako da otroke motivira in jih poleg izobraževanja tudi zabava, hkrati pa podpira tudi kreativnost otrok. Mobilno aplikacijo zasnujte kot hibridno aplikacijo z uporabo spletnih tehnologij, kar omogoča uporabo aplikacije na različnih mobilnih platformah. V diplomski nalogi raziščite tudi primernost uporabe pisala za zaslone na dotik in možnosti njihove uporabe v razviti aplikaciji.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisana Karmen Knavs, z vpisno številko **63100024**, sem avtorica diplomskega dela z naslovom:

Mobilna aplikacija za učenje črk

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom viš. pred. dr. Alenke Kavčič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 11. septembra 2015 Podpis avtorja:

Prva zahvala velja moji mentorici viš. pred. dr. Alenki Kavčič in sodelavcema z Naravoslovnotehniške fakultete: doc. Domnu Frasu in diplomantki Anji Saje. Za podporo v času študija se zahvaljujem svoji družini, še posebej staršem. Hvala Nini in Mitji za pripomočke, Evi za testiranje in Poloni za lektoriranje. Hvala vsem, ki ste me spodbujali pri diplomskem delu in mi bili v moralno oporo, zlasti fantu Luku.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	hyper text markup language	jezik za označevanje nadbesedila
CSS	cascading style sheets	seznam slogov
CLI	command line interface	vmesnik ukazne vrstice
API	application programming interface	aplikacijski programski vmesnik
SVG	scalable vector graphics	raztegljiva vektorska grafika
XML	extensible markup language	razširljivi označevalni jezik
SDK	software development kit	orodja za razvoj programske opreme

Povzetek

V diplomski nalogi je opisana izdelava hibridne mobilne aplikacije za učenje rokopisa, ki je namenjena otrokom. Poudarek razvoja je bil na operacijskem sistemu Android. Cilj je bil narediti aplikacijo za otroke, ki bo poučna, zabavna in bo omogočala kreativnost. Aplikacija Öwly je nastajala v sodelovanju z Naravoslovnotehniško fakulteto, zato sta grafična podoba in zasnova primerno prilagojeni otrokom. Predstavljene so glavne prilagoditve in značilnosti pisav, prav tako tudi uporabljene tehnologije. Uporabnik aplikacije lahko naredi več pisav. Ustvarja lahko voščilnice in pri besedilu na njih uporabi svoje pisave. Za izdelovanje hibridne mobilne aplikacije smo izbrali Cordovo. Aplikacija temelji na Javascriptu in tehnologiji HTML5, predvsem na dveh njenih elementih: canvas za risanje znakov pisave in svg za prikaz vektorskih slik. Za interakcijo z domorodno kodo je uporabljenih več vtičnikov.

Ključne besede: Cordova, hibridna mobilna aplikacija, rokopis, pisava, otroci, ustvarjanje

Abstract

The thesis describes the building of hybrid mobile application for learning handwriting targeting children. The emphasis of developing is on Android operating system. The goal was to make an application for children which is educational, fun and where creativity is possible. Application Öwly was made in cooperation with Faculty of natural sciences and engineering, therefore the graphic design and concept are sufficiently adapted for the children. The main adaptations and characteristics of writings are presented, as well as the technologies used. The application user can make more writings. He can create greeting cards and by the text on them use his own writings. For developing of hybrid mobile application we have chosen Cordova. The application is based on Javascript and HTML5 technology, particularly on two its elements: canvas for drawing the writing's signs and svg for showing the vector images. For interaction with the native code more plugins have been used.

Keywords: Cordova, hybrid mobile application, handwriting, writing, children, creating

Poglavje 1 Uvod

Mobilne aplikacije so v porastu. Vsak dan je na mobilni trg na novo vključenih več kot milijon pametnih naprav z operacijskim sistemom Android [1]. Število prenešenih aplikacij iz trgovine Google Play je približno 1,5 bilijona na mesec, iz trgovine App Store 1 bilijon in obe naraščata [2].

Otroci so postali pomembna ciljna skupina podjetij. Število tablic na družino narašča in proizvajalci izdelujejo celo tablice, prilagojene otrokom. Vsaka generacija otrok je čedalje več časa za mobilnimi napravami, povprečna starostna meja prve uporabe se znižuje. Ker so mobilne aplikacije postale pomemben dejavnik v razvoju otrok, smo se odločili za izdelavo aplikacije, ki ne bo le zabavna.

Branje in pisanje sta znanji, ki ju človek potrebuje vse življenje. Pomembno je, da se otrok (učenec) lahko osredotoči na snov oziroma vsebino in ne na samo branje. Pri pisanju besedil se lahko z boljšim znanjem rokopisa namesto na črke osredotoči na sestavo celega besedila. Na trgu so že aplikacije, ki so namenjene učenju rokopisa, vendar jih veliko ni v skladu z učnim sistemom v šoli. Pogosta slabost je tudi zaviranje kreativnosti. Naša ciljna skupina so otroci, stari od 7 do 10 let, ki se učijo brati ali že znajo.

Poučni, prvi del v aplikaciji predstavlja pisanje črk, številk in drugih znakov. Pisanje na zaslon naredi učenje bolj privlačno. Črke, številke in druge znake lahko potem uporabi v drugem delu za besedilo voščilnic. Ustvarjalnost in zabava sta podprta v obeh delih, saj otrok ni omejen s potezami, kako mora pisavo delati, elemente na voščilnicah lahko deloma ureja in sestavi poljubno besedilo. Za dodatno motivacijo je omogočeno shranjevanje in deljenje voščilnice, saj otroci radi pokažejo, kaj so naredili, in berejo besedilo s svojo pisavo.

Aplikacijo, zasnovano v sodelovanju z Naravoslovnotehniško fakulteto, smo želeli ponuditi brezplačno in čim večjemu krogu uporabnikov. Naš glavni namen je bila implementacija hibridne mobilne aplikacije s prioriteto na Androidu in narediti uporaben izdelek. Pri razvoju smo se trudili upoštevati tudi posebnosti za okolje iOS.

V nadaljevanju najprej opišemo značilnosti pisav in otroške aplikacije, nato hibridne mobilne aplikacije, Cordovo in uporabljene tehnologije, zatem samo izvedbo. V sklepu argumentiramo ustreznost uporabe Cordove in podamo možnosti za izboljšavo aplikacije.

Poglavje 2 Pisanje v mobilni aplikaciji za otroke

2.1 Rokopis

Je besedilo, napisano z roko. Raven rokopisa se meri s čitljivostjo in hitrostjo pisanja. Z vajo lahko posameznik hitreje razvije podzavestno pisanje črk. V tehnološkem svetu postaja pomembno učenje preko zaslona, vendar to ne more nadomestiti pravega pisanja s pisalom na papir, tudi če piše s pisalom za zaslone na dotik. Prednosti učenja na mobilnih napravah so večja privlačnost ter dodatne funkcionalnosti, kot so brisanje, povrnitev izbrisane poteze, barve, animacije, zvok.

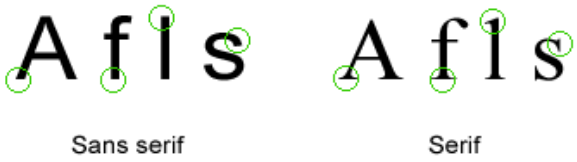
2.1.1 Tipografija

Pisava je sistem znakov, ki predstavljajo neko vsebino. Tipografija je orodje pisave za vizualno komuniciranje. Kot je govor zvočna oblika jezika, je tipografija (pisava) njegova vizualna podoba. Njena osnovna naloga torej je, da zapisuje jezik [3].

Če je besedilo čitljivo, še ne pomeni, da je tudi berljivo. Čitljivost pomeni, da ga lahko interpretiramo. Berljivost kombinira čustveni vpliv grafike s količino truda, potrebnega za branje – pojem se navezuje na vprašanje, ali besedilo hočemo prebrati in ne, če ga lahko preberemo [4]. Dobro oblikovana tipografija mora torej biti jasna, nazorna, čitljiva in človeku prijazna komunikacija. V ožjem pomenu izraz uporabljamo tudi za črkovno vrsto. Tipografija je dobila velik pomen z razvojem računalnikov oziroma digitalnih pisav [3].

Glede na oblike in izhodišče nastanka ločimo več skupin tipografskih črkovnih vrst (pisav):

- črke s serifi (z različno oblikovanimi zaključki),
- črke brez serifov (sans serif, brez zaključkov).



Slika 2.1: Primer pisave brez zaključkov (levo) in njeno nasprotje (desno).[1]

Na sliki 2.1 je primer obeh pisav. Posamezna črka je po navadi izdelana v več različicah – pokončni (normalna), ležeči (kurzivna), polkrepki in krepki –, ki skupaj tvorijo družino pisave [3].

2.1.2 Pisava v aplikaciji

Pomembna lastnost tipografije na zaslonu v primerjavi s tiskano je, da se bralci soočajo z dvema bralnima okoljema: s fizičnim prostorom (in njegovo osvetljenostjo) in napravo. Uporabnik lahko bere zunaj na soncu, v sobi z zastrtimi okni ali v temi pred spanjem. Najboljša rešitev je, da se tipografija dobro obnese v vseh situacijah, ne glede na velikost zaslona [4].

Pri drobnem besedilu je lažje brati majhno neserifno pisavo, pri tiskanih sporočilih pa večjo in serifno pisavo, ker zaradi zaključkov deluje bolj povezano. Na zaslonu je lažje berljiva in jasnejša pisava brez serifov [3]. Ker je aplikacija namenjena otrokom, ki se učijo brati in pisati, je pomembno, da je pisava dobro berljiva in da so črke oblikovane podobno, kot se jih učijo v šoli. Na sliki 2.2 je označen karakter črk. Pravilna pisava ima ves čas enako višino male črke.

Slika 2.2: Karakter črk (pisava Bodoni) [3].

Črki a in g ter številke 1, 3 in 4 so znaki, ki so pogosto drugačno oblikovani od rokopisnih. Neskladni črki sta vidni tudi na primeru na sliki 2.2. Pisave z višjim srednjim pasom (višino male črke) veljajo za bolj berljive. V aplikaciji smo uporabili dve različni črkovni družini. Za

[1] http://www.naturprint.com/wp-content/uploads/2014/02/diferencia-serif-sansserif.gif

predlogo v ozadju pri risanju črk in kot podoba znaka, ki ga mora uporabnik ustvariti, je uporabljena *AG schoolbook™ Regular*, za ostalo besedilo *londrina solid Regular.*

2.2 Uporabniški vmesnik in uporabniška izkušnja

Navigacija v aplikaciji mora biti preprosta, ne sme vsebovati preveč podmenijev. Imeti mora smiselno postavitev elementov. Za otroke so bolj primerne ikone s sliko akcije kot ikone z besedilom. Pomembna je privlačna tematika. Glavni lik aplikacije je sovica, ki ponazarja modrost. Ciljna skupina otrok ima rada barvni uporabniški vmesnik, same slike so lahko že z več detajli. Izbrane barve in grafike ustrezajo obema spoloma.

Otrok je zahteven uporabnik, saj zahteva visoko zmogljivost in odzivnost aplikacije. Uporabniški vmesnik mora biti zasnovan tako, da je samoumeven otroku.

2.3 Pisala za zaslone na dotik

Poznamo pasivna (angl. capacitive stylus) in aktivna pisala za zaslone na dotik. Mobilne naprave zaznavajo pasivna pisala enako kot prst. Aktivna pisala lahko omogočajo različne debeline pisanja glede na pritisk na površino zaslona, radiranje s klikom na pisalu ali obračanjem pisala, odboj dlani (angl. palm rejection) ... V Androidu je bila podpora zanje dodana v verziji 4. Glede na povezovanje z napravo ločimo aktivna pisala na dve skupini [5].

Prva skupina se povezuje preko tehnologije digitalizatorja podjetja Wacom, kar pomeni, da mora biti podprta od proizvajalca, oziroma vgrajena v zaslon naprave, da lahko dela. Nadalje morajo biti aplikacije spisane tako, da informacije, povezane s pisanjem, zaznavajo in uporabljajo.

Druga skupina se povezuje preko tehnologije Bluetooth 4.0 (Bluetooth Low Energy), ki je z Androidom podprta le na novejših napravah, na napravah s sistemom iOS pa od vključno iPhone 4S dalje. Za aktivno delovanje mora aplikacija omogočati povezovanje (angl. pairing). Pisala je potrebno polniti in imajo svojo strojno-programsko opremo (angl. firmware), kar pomeni, da ima vsak proizvajalec pisala svoj SDK, ki ga je potrebno vključiti [6]. Čeprav je zunaj čedalje več naprav Android s tehnologijo Bluetooth 4.0, je SDK za Android še v razvojni fazi.

2.3.1 Aktivno pisalo v aplikaciji

Čeprav bi bil z uporabo aktivnega pisala otrok bliže pravi izkušnji pisanja, se za podporo aktivnemu pisalu v aplikaciji nismo odločili iz več razlogov:

- SDK za našo ciljno platformo Android je še v razvojni fazi,
- težko bi bilo podpreti več aktivnih pisal,
- aktivna pisala uporablja majhno število uporabnikov mobilnih naprav,
- aktivna pisala so zaenkrat še precej draga (najcenejša blizu 20€, najdražja blizu 100€),
- pisanje s prstom na zaslon je za otroke intuitivno.

Pri podpori aktivnega pisala, ki se povezuje preko bluetootha, bi morali v aplikacijo vključiti dodaten gumb, samo povezovanje bi verjetno morali izvesti starši. Pri ustvarjanju s pisalom bi morali podpreti debelino pisanja glede na pritisk na površino zaslona in odboj dlani, saj otroci težko pišejo, ne da bi se naslanjali, drugače tako pisanje ne bi bilo naravno. Če bi hoteli, da otroci res pišejo pravilno, bi morali prikazovati vrstni red potez za vsako črko, skladen s šolskim načinom učenja, ter ob vsaki narejeni potezi preverjati položaj roke in pravilnost potega. Omejenost na ta način bi zavirala kreativnost, vendar bi lahko vključili možnost izbire prostega pisanja in pisanja po pravilih. Morali bi paziti na ažurnost vrstnega reda potez s šolskim načinom učenja.

Težko ali celo nemogoče bi bilo simulirati trenje pri pisanju na papir, različne vrste svinčnika (mehki, trdi) ter nadzorovati pravilno držo. Učenje pisanja na papir ne bi mogli nadomestiti tudi zaradi vpliva dveh svetlobnih okolij (zaslon in fizični prostor).

2.4 Objava aplikacij za otroke

Otroci so občutljiva ciljna skupina, zato so zanje pri objavi aplikacije dodani posebni členi[2], ki se nanašajo na varstvo podatkov. Aplikacija mora vsebovati zanje primerno vsebino, pri povezovanju izven aplikacije je potrebna posebna potrditev staršev (pravilo je bilo sprejeto zaradi primerov, ko so otroci kupovali brez vednosti staršev). Reklame niso prepovedane, morajo pa biti usmerjene v otroško ciljno skupino. Naša aplikacija ne zahteva nobenih osebnih podatkov otroka, reklam in nakupov ne vsebuje. Pri deljenju voščilnice so lahko deloma razkriti osebni podatki (odvisno od besedila, ki ga uporabnik sestavi, in od tega, kdo je lastnik naprave). Smiselno bi bilo dodati možnost, da starš lahko onemogoči delitev na

[2] https://developer.apple.com/app-store/review/guidelines/#kids-category - posebni členi za objavo v trgovini App Store, ki veljajo za otroke. Podobno velja tudi pri objavi za druge operacijske sisteme.

socialnih omrežjih. Za objavo v trgovini Apple Store bi morali vključiti poseben pravilnik o zasebnosti [7].

Poglavje 3 Hibridne mobilne aplikacije

3.1 Delitev mobilnih aplikacij

Mobilne aplikacije delimo v domorodne, hibridne in spletne. Pri domorodni izrabljamo vse sposobnosti naprave in operacijskega sistema z minimalnim performančnim presežkom. S postavitvijo spletne aplikacije lahko močno zmanjšamo razvijalni čas in stroške. Hibridna aplikacija združuje lastnosti obeh prejšnjih aplikacij – ima skupno kodo za več različnih operacijskih sistemov in enake zmožnosti kot domorodna.

Z vidika uporabnika aplikacije je pomembno, da je njegova uporabniška izkušnja enako dobra oziroma dovolj blizu kot pri uporabi domorodnih aplikacij. Ker je naša aplikacija zasnovana tako, da je izgled enak na vseh operacijskih sistemih (podobno kot igra) in ne vsebuje težjih domorodnih funkcionalnosti, prehodov in animacij, smo ocenili, da je ustrezna za izdelovanje s hibridno mobilno tehnologijo [8].

3.2 Apache Cordova

Je brezplačno odprtokodno ogrodje za izdelovanje mobilnih aplikacij za več platform. Omogoča izdelavo hibridnih aplikacij, kar pomeni, da aplikacijo sestavljajo tako domorodne kot spletne tehnologije. Spletna aplikacija teče v lupini domorodne aplikacije, v notranjem brskalniku.

Cordova tako obsega:

- domorodno kodo za vsako platformo, ki prikazuje HTML5 aplikacijo na napravi preko spletnega pogleda (angl. Web view): na Androidu WebView, na iOSu UIWebView ...,
- zbirke vmesnikov API: vsak API je iz ene knjižnice Javascript in (ponavadi) več domorodnih implementacij,
- orodja, s katerimi lahko ustvarjamo projekte, urejamo vtičnike, gradimo in testiramo aplikacije na napravi ali simulatorjih in posnemovalnikih [9].

3.2.1 Primerjava z Unity

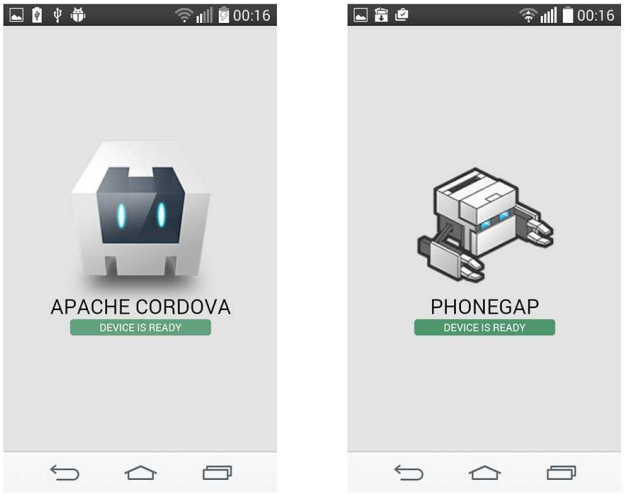
Preden smo se odločili za uporabo ogrodja Cordova pri implementaciji naše aplikacije, smo pregledali še nekatera alternativna ogrodja za izdelavo hibridnih aplikacij. Tudi ogrodje Unity, ki prevede celo spisano kodo v domorodno za vsako platformo, ki jo podpiramo, bi bilo lahko ena od alternativ. Vendar pa ima ogrodje Unity kar nekaj slabosti, zaradi katerih se nismo odločili zanj:

- težja izdelava vtičnikov, če bi bili potrebni – v brezplačni verziji Unityja v času odločanja domorodni vtičniki niso bili podprti (v času pisanja so v Unity verziji 5 podprti tudi v brezplačni verziji),
- večja izraba baterije – ker pogon za igro nima dogodkovno vodene arhitekture,
- manjša primernost glede na tip aplikacije – ne bi uporabljali pravih Unity zmožnosti, kot so 3D svet, zvok ...

Ker Cordova aplikacija poteka v lupini, je izvajanje nekoliko počasnejše, vendar ne toliko, da bi smelo vidno vplivati na uporabniško izkušnjo (nimamo težkega računanja in animacij). Spoznavanje z novo tehnologijo smo sprejeli kot dodaten plus.

3.2.2 Primerjava s Phonegap

Adobe Phonegap je implementacija Apache Cordove z dodatki. V samem jedru je vsebina (ponavadi ne zadnje verzije) Cordove, zraven vsebuje še storitev za izgradnjo aplikacij Build Phonegap [9][10]. Orodja Phonegapa, ki jih uporabljamo v ukazni vrstici za urejanje projektov, so Cordovina (le drugače poimenovana) z nekaj dodatnimi. Če ustvarimo novo aplikacijo s Cordovo ali Phonegapom, dobimo praktično enaki aplikaciji – prikazani sta na sliki 3.1.

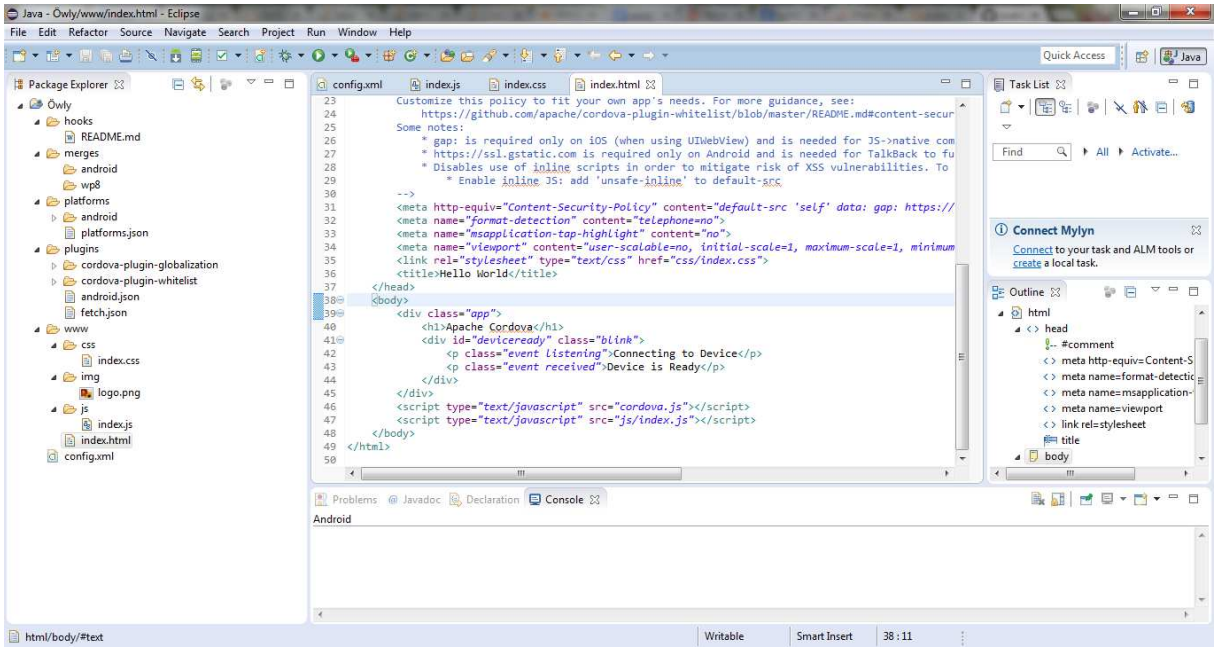


Slika 3.1: Začetna aplikacija Cordove in Phonegapa.

Tudi dokumentaciji za obe ogrodji izgledata zelo podobni. Na svetovnem spletu se imeni pogosto uporabljata, kot da obe pomenita Cordovo.

3.2.3 Zgradba Cordovinega projekta

Cordova aplikacija je sestavljena iz štirih glavnih map: hooks, platforms, plugins in www.



Slika 3.2: Zgradba začetne Cordovine aplikacije, ustvarjene preko vmesnika ukazne vrstice v okolju Eclipse z vtičnikom Thym

V mapi www imamo kodo spletne aplikacije, ki je skupna več platformam.

Ko dodamo ciljni operacijski sistem, se v mapo platforms doda ustrezna domorodna koda. Ob vsakem grajenju se koda posodobi, skupni viri (HTML, CSS, Javascript) na vrhu aplikacije v www se kopirajo v mapo www ciljnih platform na nižjem nivoju. Vsaka dodana ciljna platforma ima svojo ločeno mapo.

V mapi plugins so datoteke z domorodno kodo za vsak dodan vtičnik. S Cordovo verzija 3.0 je bil vsak API razdeljen na več vtičnikov. Postala je nekakšen hibridni vsebovalnik (angl. container). Tako v osnovi vsebuje le glavne dogodke [9]. Na sliki 3.2 smo že dodali vtičnik za globalizacijo, vtičnik whitelist je vključen že na začetku.

Mapa hooks (*hook* v slovenščini pomeni trnek, kljuka) predstavlja prostor za posebne skripte, katere lahko dodajo aplikacija in razvijalci vtičnikov ali programer sam, da prilagodi cordovine ukaze. Podprtih je več različnih vrst, našteli bomo le nekaj primerov: *after_build, after_compile, after_clean, after_docs, after_emulate, after_platform_add, after_platform_rm, after_plugin_install* ... (vsak ima tudi ustrezno različico z *before_* predpono – na primer *before_build*). Lahko je spisan v Javascriptu ali v skriptu ukazne lupine (angl. shell script) [11].

Mapa merges ni ustvarjena privzeto. V njej lahko definiramo datoteke, specifične za določene platforme – na primer index.html ali CSS različen za iOS in Android.

3.2.4 Ukazna vrstica Cordova in osnovni dogodki

Vmesnik ukazne vrstice Cordova je zgrajen iz kode Javascript, ki je izpostavljena preko izvajalnega okolja (angl. runtime engine) Node Javascripta. Zato lahko teče povsod, kjer je dosegljiv *Node.js*. Uporablja proces lenega nalaganja (angl. lazy loading) – datoteke projekta so prenešene, ko oziroma če je potrebno. Ob ustvarjanju projekta – z ukazom *create* – vmesnik ukazne vrstice prenese privzet HelloWorld projekt, ga razširi (unzip) in kopira v ustrezno mapo [9]. Ukazi vmesnika ukazne vrstice so: *help, create, platform, plugin, prepare, compile, build, emulate, run, serve* [11].

Osnovni dogodki (angl. core events) oziroma življenjski dogodki (angl. lifecycle events), ki so na razpolago že v privzetem projektu, so našteti in pojasnjeni v tabeli 3.1. Najpomembnejši je *deviceready*, saj lahko po njegovi sprožitvi varno dostopamo do domorodnih programskih vmesnikov [10].

Osnovni dogodek	**Razlaga**
deviceready	naprava je pripravljena, Cordova naložena
pause	aplikacija je potisnjena v ozadje
resume	aplikacija je priklicana iz ozadja
menubutton	klik na mehanski ali programski meni (Android, BlackBerry)
searchbutton	klik na iskanje (le Android)
backbutton	klik na gumb nazaj
online	naprava se poveže z internetom
offline	izguba internetne povezave
startcallbutton	klik na gumb za klicanje (BlackBerry)
endcallbutton	klik na gumb za prekinitev klica (BlackBerry)
volumedownbutton	klik na gumb za zmanjšanje glasnosti (BlackBerry)
volumeupbutton	klik na gumb za povečanje glasnosti (BlackBerry)

Tabela 3.1: Seznam osnovnih dogodkov Cordove.

Za uporabo dogodkov, vezanih na status baterije (*batterycritical, batterylow, batterstatus*), je od verzije 3.0 potreben vtičnik. V iOSu se koda, vezana na *pause,* dejansko izvede šele ob priklicu iz ozadja. Alternativa za *resume* je *active*, za *pause* je *resign*. Vezana sta na gumb za zaklepanje [11].

3.2.5 Uporabljeni vtičniki v aplikaciji

V aplikaciji smo uporabili 5 vtičnikov. *Whitelist* je že privzeto naložen varnostni model, ki nadzoruje dostop do zunanjih domen. Za shranjevanje podatkov v podatkovno bazo *sqllite* smo uporabili vtičnik *Cordova sqllite storage plugin (io.litehelpers.cordova.sqlite)*. Za zaznavanje prikaza in skritja tipkovnice smo uporabili vtičnik *Keyboard (com.ionic.keyboard)*. Pri določanju lokalizacije smo si pomagali z vtičnikom *Globalization (org.apache.cordova.globalization),* pri deljenju ustvarjene voščilnice pa s *Socialsharing (nl.x-services.plugins.socialsharing)*.

Poglavje 4 Spletne tehnologije in uporabljena orodja

4.1 HTML5 in CSS3

HTML5 je W3C (World Wide Web Consortium) standard za strukturiranje in predstavitev vsebine na svetovnem spletu, naslednik HTML 4.01, XHTML 1.0 in XHTML 1.1. Vključuje vrsto novih elementov in atributov, ki pomagajo pri zgradbi moderne spletne strani: nove semantične elemente, Forms 2.0 (novi atributi za značko input), stalno lokalno shrambo (angl. persistent local storage), tehnologijo WebSocket, podporo za dogodke, poslane s strežnika, canvas, svg, avdio in video, geolokacijo, mikro podatke (angl. micro data), povleci-in-spusti (angl. drag-and-drop) [12].

CSS (Cascading Style Sheets) je oblikovni jezik, ki poenostavlja proces dodajanja in urejanja izgleda spletnih strani. Z njegovo pomočjo prihranimo čas in lažje vzdržujemo strani, saj nam ni treba določati stilnih atributov za vsak element dokumenta posebej, ker lahko strnemo pravila s selektorji in razredi. Hkrati poskrbimo tudi za hitrejše nalaganje s strežnika in kompatibilnost prikaza vsebine za različne naprave. W3C je izdal prvo priporočilo za CSS3 že junija 1999 (to pomeni, da so razvite specifikacije stabilne in so jih pregledali člani konzorcija). Med novosti spadajo zaobljeni vogali, okviri iz vzorca slik, gradient, sence, transformacije v 2D in 3D, tranzicije, animacije, medijske poizvedbe (angl. media queries), imenski prostor (angl. namespace), dodatna pravila za selektorje, tekst, barve, pisavo (angl. font-face), ozadja, uporabniški vmesnik, možnost oblikovanja več stolpcev in določanja velikosti elementa (angl. box-sizing) [12].

V aplikaciji smo pisavo, ki smo jo ustvarili v štirih različnih formatih (.eot, .svg, .ttf, .woff), vključili s pomočjo selektorja font-face. Za lažje delo s tabelami smo vključili sistem Bootstrap grid.

4.2 Javascript

Organizacija ECMA definira Javascript kot lahek, odprtokoden in tolmačen (angl. interpreted) programski jezik, ki je podprt na več platformah. Najpogosteje je uporabljen na spletnih

straneh kot skripta na strani odjemalca in omogoča interakcijo z uporabnikom in dinamičnost vsebine. Prednosti Javascripta so predvsem takojšnje povratne informacije, manj interakcije s strežnikom, povečana interakcija z uporabnikom in bogatejši vmesniki. Njegova omejitev je, da zaradi varnosti ni možno brati/pisati datotek, ni večnitenja ali izvajanja na več procesorjih hkrati [12]. Z uporabo skripte v ozadju (tako imenovani *web worker*) lahko intenzivne računske naloge izvajamo brez blokiranja uporabniškega vmesnika.

Pri gradnji hibridnih mobilnih aplikacij s Cordovo je priporočeno, da je čim več funkcionalnosti strnjenih na eni strani, saj prehajanje med stranmi lahko povzroči težave s spominom ali počasnejšo interakcijo. Javascript je zato še bolj pomemben, saj omogoča dinamično nalaganje [9].

4.2.1 Uporabljene knjižnice

V aplikacijo smo vključili naslednje knjižnice:

- jQuery (*jquery-1.11.3.min.js*) za lažjo kompatibilnost izvajanja Javascripta na različnih operacijskih sistemih,
- iScroll (*iscroll-lite.js*) za boljšo uporabniško izkušnjo pri elementih, kjer je potrebno premikanje v vidni del zaslona,
- *jquery.animate-enhanced.min.js* za lepše animacije,
- *html2canvas.js* za generiranje slike dokumenta.

4.3 Canvas

Canvas je element HTML5, ki ponuja možnost risanja grafike z Javascriptom. Je bitno pravokotno platno, odvisno od resolucije. Z njim lahko rišemo grafe, ustvarimo kompozicijo slike, delamo animacije in transformacije ... Nekateri razvijalci so eksperimentirali s svojimi vmesniki 3D canvas API, vendar je zaenkrat standardiziran le 2D. Vsak canvas vzdržuje svoje stanje in ima svoj kontekst za risanje (angl. drawing context), v katerem so definirane metode in lastnosti za risanje. Za risanje ravne črte se najprej premaknemo v začetno točko z metodo moveTo(x,y), nato določimo končno točko z lineTo(x,y) in potegnemo črto z ukazom stroke(). Možno je tudi shraniti stanje in ga obnoviti. To pomeni, da lahko kasneje ponovno naložimo transformacije, območje rezanja in lastnosti oziroma stil risanja [13].

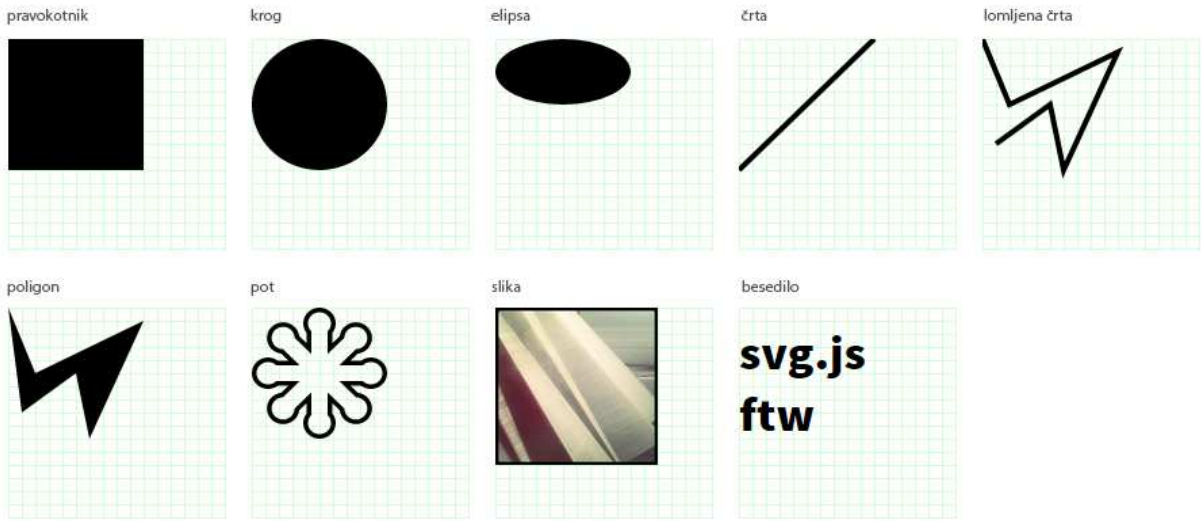
Ustvarimo lahko osnovne like, ravne in krive črte, besedilo, gradient, slike, vzorce in sence. Za enostavnejše igre na element canvas dodamo odzivanje na dogodek (eventListener) za klik

in/ali dotik in tako poskrbimo za interakcijo z uporabnikom. Z vsako akcijo vsebino pobrišemo in na novo narišemo.

4.4 SVG

SVG (angl. scalable vector graphics) je jezik, ki je osnovan na jeziku XML in definira raztegljivo vektorsko (objektno, predmetno) grafiko. Uporablja se za risanje dvodimenzionalnih vektorskih slik in je W3C standard. Slika SVG s povečevanjem in rotiranjem ne izgublja na kvaliteti, podpira interaktivnost in animacijo. Lahko vključuje rastrske (bitne) slike. SVG je vključen v objektni model dokumenta HTML (DOM). Ker je osnovan na jeziku XML, so slike SVG lahko indeksirane, kompresirane, skriptirane in se jih lahko spreminja z vsakim urejevalnikom besedila [12].

V rastrski grafiki je slika predstavljena z množico obarvanih točk (piksli), v vektorski z matematičnimi objekti ali predmeti. Na sliki 4.1 so elementi SVG – prvih 6 elementov je osnovnih. Slabost tekstovnega formata je opis težje matematično definiranih slik (fotografije so zato vedno rastrske), binarne rastrske slike so lahko zato tudi manjše od vektorskih.



Slika 4.1: Elementi grafike SVG.[3]

4.5 Omejitve mobilnih brskalnikov

Ker Cordova teče v spletnem pogledu, je pomembno, da upoštevamo omejitve novejših mobilnih brskalnikov, predvsem tistih, ki so naloženi na napravah s sistemi Android 4 in 5 (z

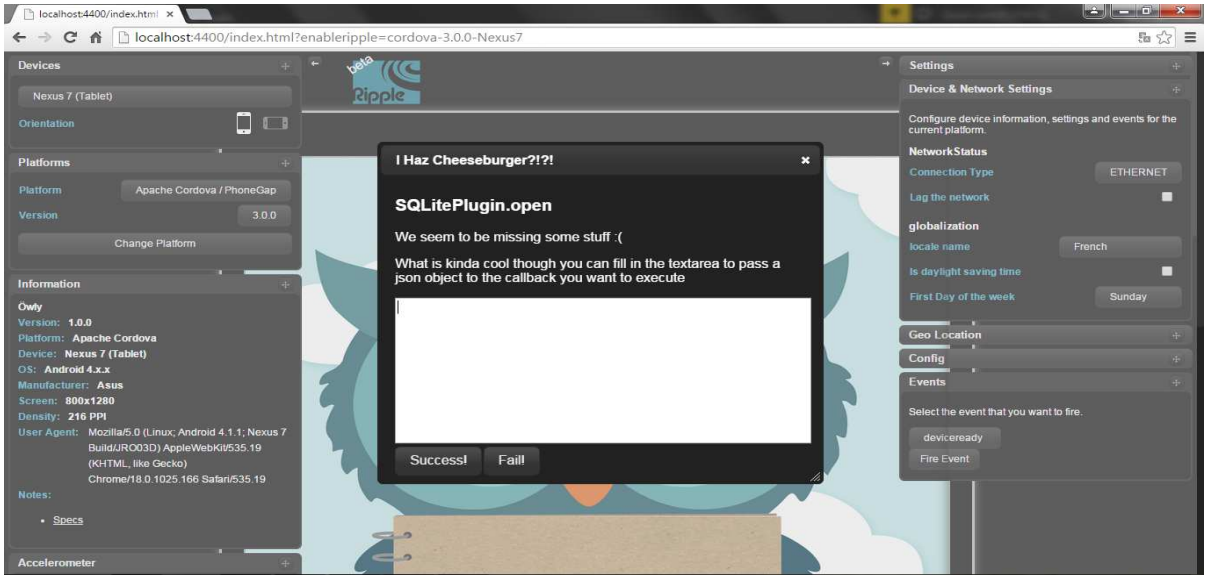
[3] http://svgjs.com/test/

verzijo 4.4 aplikacija teče v brskalniku Chromium), iOS 7 in 8, Windows Phone 8. Vsi imajo dobro podporo za HTML5, razen brskalnika Opera Mini. Slednja ima tudi slabo podporo za CSS3; ta je v drugih brskalnikih dobra, samo Internet Explorer ima težave s 3D transformacijami. V vseh mobilnih brskalnikih je mogoča uporaba elementov inline-svg in canvas. Na sistemih iOS, Android 4.1 in Android 4.3 transformacije CSS ne delujejo na elementih SVG, kar se da nadomestiti tako, da spreminjamo atribut transform na objektih. Skripte Web worker lahko uporabljamo na najnovejših različicah (na Androidu od vključno verzije 4.4), web storage na vseh razen Opera Mini [14].

Na Androidu je uporaba HTML5 pravzaprav mogoča že v verziji 2.3, vendar je podpora precej slaba. Izvajanje hibridne aplikacije je na starejših napravah lahko počasno.

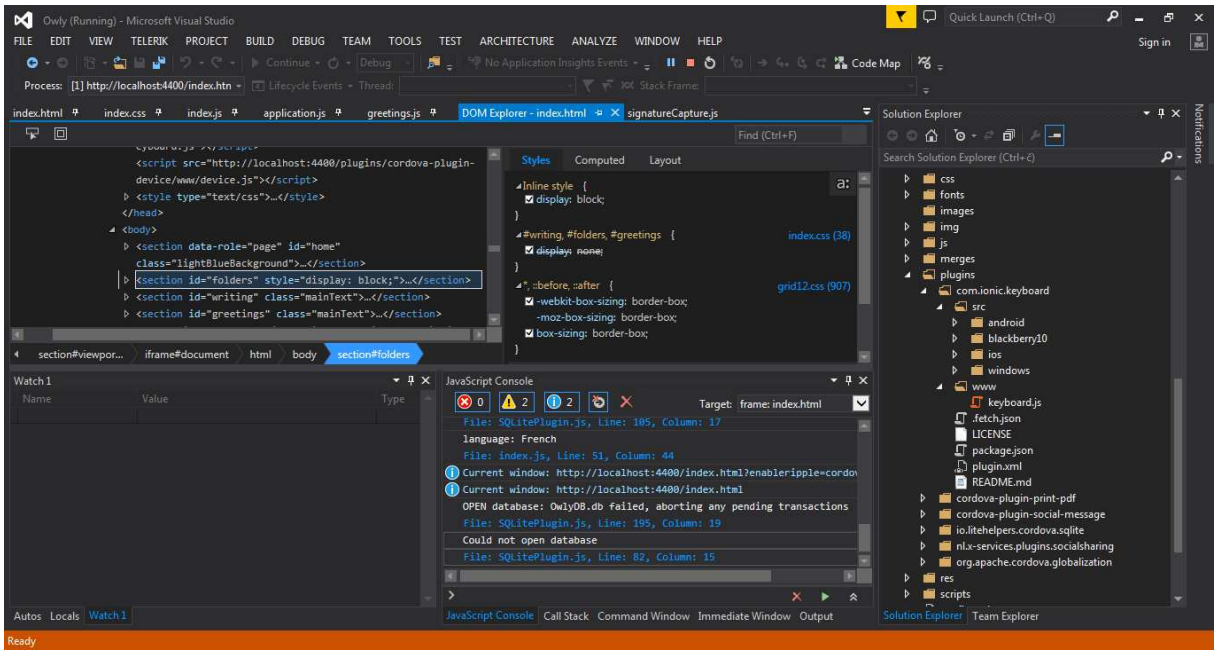
4.6 Razvojna in testna orodja

Za razvoj aplikacij z orodjem Cordova ne obstajajo posebna namenska razvojna okolja. Kodo lahko urejamo celo v urejevalnikih besedil in jo nato zgradimo ter naložimo s pomočjo ukazne vrstice, vendar je na tak način aplikacijo težko razvijati. Pri delu za vizualno urejanje aplikacije smo uporabljali posnemovalnik Ripple Emulator, ki posnema več mobilnih platform. Naloži se ga kot vtičnik za brskalnik Chrome in nato tudi teče v njem. Posnemovalnik omogoča proženje dogodka *deviceReady*, obračanje zaslona, simulira domorodna vtičnika za geolokacijo in globalizacijo. Na sliki 4.2 je prikazano posnemovalno okolje in kaj se zgodi, kadar testiramo druge domorodne vtičnike.



Slika 4.2: Ob klicu domorodnega vtičnika Ripple ponudi možnost pošiljanja argumenta v objektnem zapisu Javascripta (JSON).

Ker smo hoteli imeti lepši pregled nad kodo, aplikacije nalagati neposredno na napravo in jih testirati, smo raziskali podporo v večjih razvijalnih okoljih[4]. Najprej smo razvijali v Eclipsu z vtičnikom Thym (na sliki 3.2), ker pa nismo bili zadovoljni z razhroščevanjem, smo nato nadaljevali v Visual Studiu 2013 z vtičnikom Visual Studio Tools for Apache Cordova. Z njim lahko aplikacijo naložimo neposredno na napravo in nato spreminjamo objektni model dokumenta HTML in atribute CSS ter vidimo takojšen odziv na zaslonu, kot kaže slika 4.3. Omogočeno je tudi postavljanje prekinitvenih točk in sledenje izvajanju kode Javascript.



Slika 4.3: Razhroščevanje Cordova aplikacije v Visual Studiu na posnemovalniku Ripple.

Ugotovili smo, da posnemovalniki niso čisto zanesljivi za testiranje prikaza aplikacije in imajo lahko težave s prikazom lastnosti CSS3 (na primer računanje višine naprave je drugačno, posnemovalnik za Android, zagnan z Visual studiom, pa ni prikazoval slike, definirane kot ozadje).

Ikone smo izdelali s pomočjo Adobe Ilustrator Portable in Adobe Photoshop. Razrede objektov svg smo določili s programom Inkscape 0.91.

[4] Razvoj Cordova aplikacij je podprt v okoljih Eclipse, NetBeans, Visual Studio, Webstorm.

Poglavje 5 Implementacija in predstavitev aplikacije

Aplikacijo lahko razdelimo na dva glavna dela. Prvi del je namenjen predvsem učenju oziroma vaji pisanja, kjer uporabnik riše posamezne znake pisave. Drugi del je namenjen uporabi ustvarjene pisave, torej zabavi in kreativnosti. Podrobneje aplikacijo razdelimo na štiri strani: začetno domačo stran, stran s shranjenimi pisavami, stran za ustvarjanje pisave in stran za ustvarjanje voščilnic. Sova kot glavni lik povezuje strani, otroke motivira in naredi aplikacijo bolj zanimivo. Grafika je umirjena, prepojena z motivi iz narave.

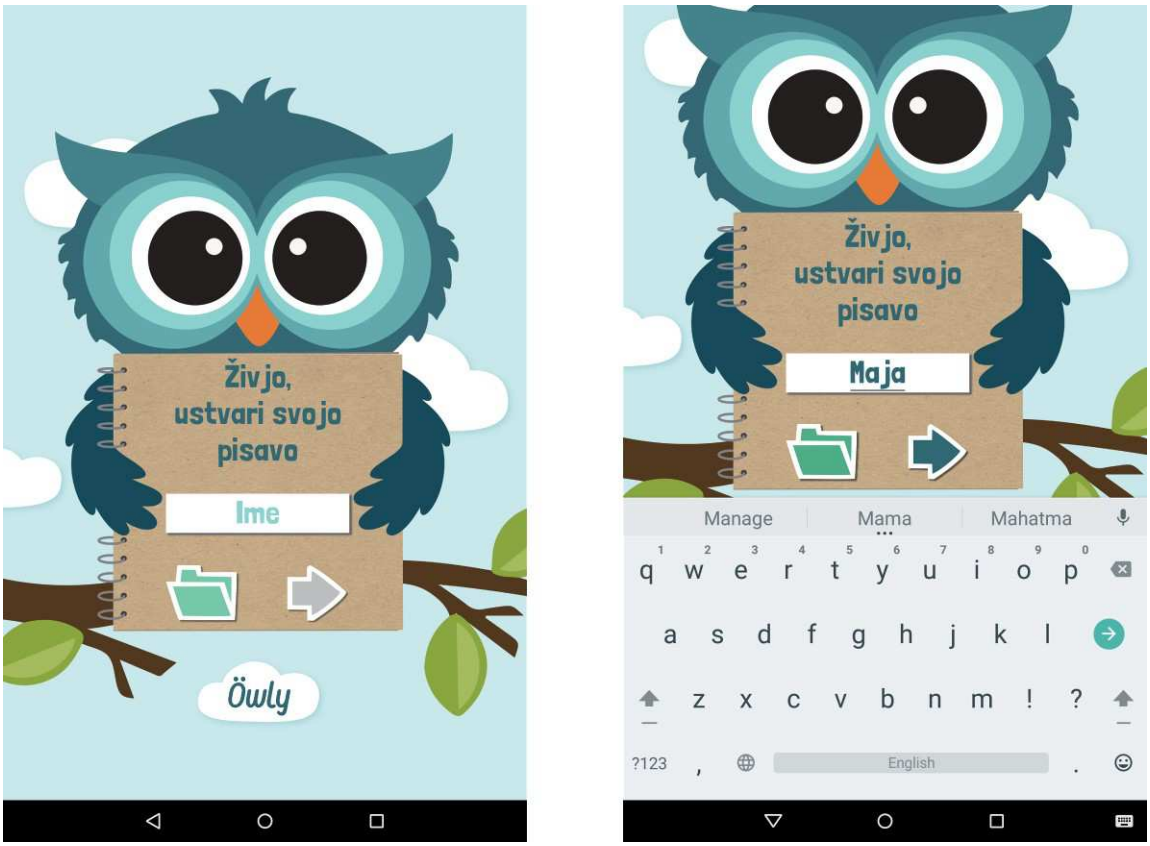
Pri delu smo sodelovali z diplomantko Naravoslovnotehniške fakultete, ki je pripravila tudi grafični koncept aplikacije. Skupaj smo predebatirali zastavljeno zasnovo aplikacije, predlagali vsak svoje ideje ter določili končne funkcionalnosti.

5.1 Domača stran

Aplikacija se začne z nalagalnim zaslonom, ki je prikazan na sliki 5.1. Android aplikacije so sicer lahko tudi brez, iOS pa ne, saj v trgovini App Store drugače zavrnejo aplikacijo. Zaradi usklajenosti smo se odločili, da ga bomo vključili v razvoju za vse platforme.

Slika 5.1: Nalagalni zaslon v aplikaciji.

Nato se prikaže prvi zaslon, to je domača stran, vidna na sliki 5.2 – 'stran' tu ni mišljena kot fizična spletna stran, saj so pravzaprav vsi zasloni del ene same; posamezne sekcije dokumenta so ustrezno prikazane oziroma skrite.



Slika 5.2: Domači zaslon, levo začetni, desno ob vnosu in z že narejeno pisavo.

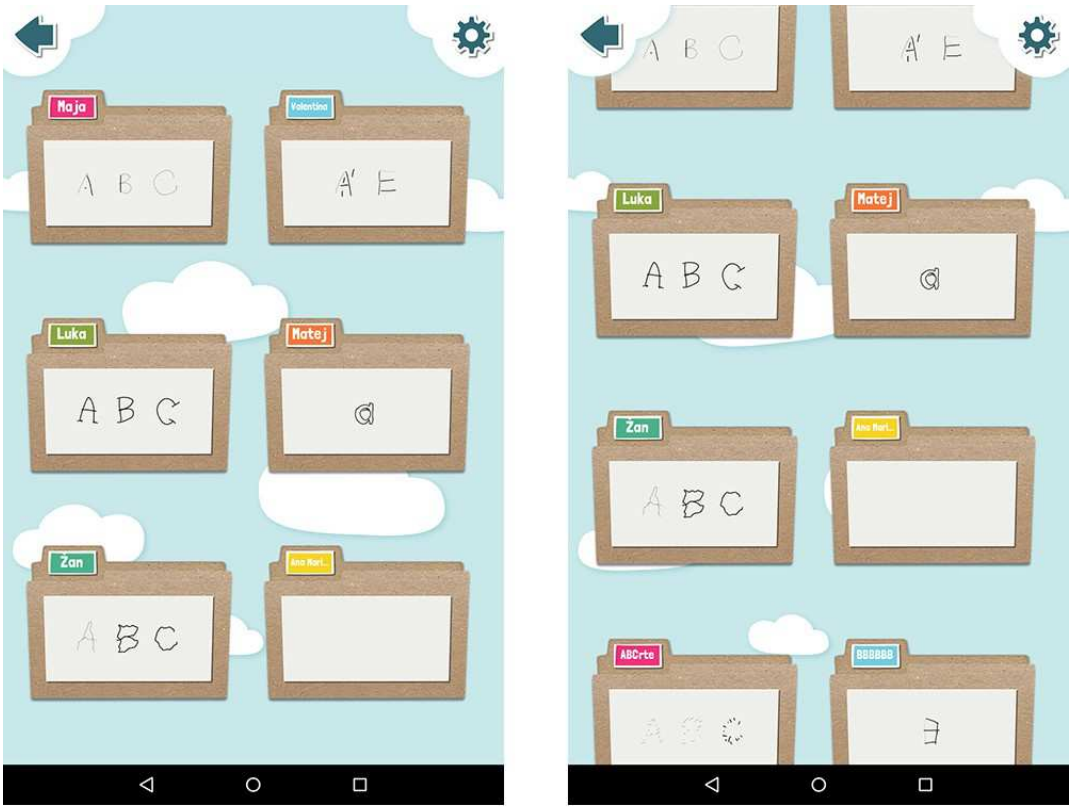
Uporabnik lahko tu vnese novo ime pisave ali svoje lastno ime. V ozadju se naloži Cordova in vtičniki. Ob dogodku *deviceReady* se preveri, ali pisave že obstajajo, in preveri lokalizacija, ki je nastavljena na napravi. Če obstaja vsaj ena pisava, se mapica, ki vodi naprej do zbirke pisav, pobarva in omogoči akcijo. Z vtičnikom za globalizacijo preverimo lokalizacijo sistema in ustrezno nastavimo jezik na slovenščino ali angleščino ter shranimo v lokalno shrambo (angl. localstorage). To potem upoštevamo pri prikazovanju črk na strani ustvarjanja pisav in na mapicah na strani s shranjenimi pisavami. Ko uporabnik vnaša ime, se gumb za naprej na stran za pisanje (puščica) obarva in sova s knjigo dvigne, zato da ima uporabnik pregled in možnost za izbiro vseh akcij – slika 5.2 desno. Možno je samo vnašanje črk in številk.

Aplikacijo smo najprej razvijali z vidno statusno vrstico in takrat smo lahko pridobili višino tipkovnice preko dogodka Javascripta *resize*, ker se dejansko spremeni velikost zaslona Cordove. Ko smo spremenili aplikacijo v celozaslonsko, pa se velikost več ne spreminja in višina tipkovnice ostaja neznana. Edina možna rešitev je uporaba vtičnika. Zamikanje ozadja

smo najprej naredili samo z uporabo knjižnice jQuery, vendar animacija ni bila tekoča. Zato smo dodali knjižnico, ki animacije avtomatsko prevede v animacije CSS3. Te tečejo s pomočjo strojne pohitritve.

5.2 Stran s shranjenimi pisavami

Na to stran lahko pridemo samo z domače strani. Vsaka mapa predstavlja pisavo, zgoraj ima napis imena pisave, v osrednjem belem prostoru so vidne prve tri narejene črke pisave po abecednem vrstnem redu. Na sliki 5.3 so vidne vse barve ozadja napisov map, ki se potem ponavljajo, in spreminjanje velikosti pisave glede na število črk v napisih (ime Maja ima večjo pisavo kot Valentina).



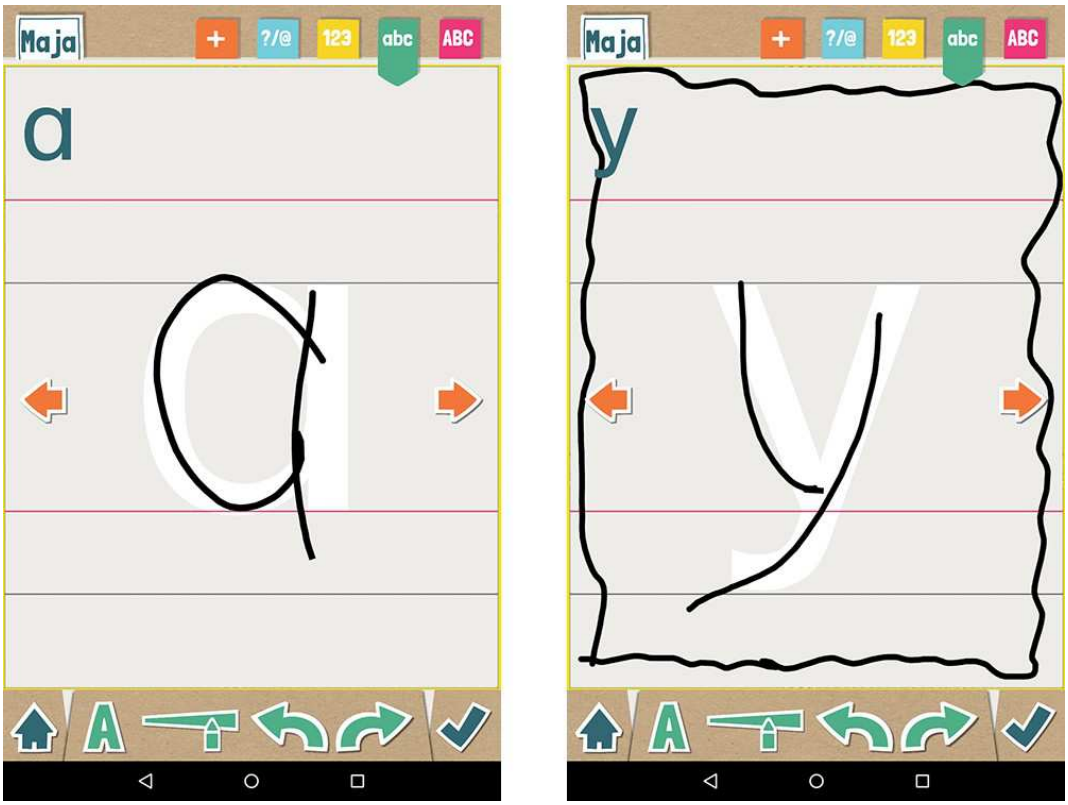
Slika 5.3: Levo je vidnih šest možnih barv napisov, na desni videz po premiku seznama.

Zgoraj nad mapami je meni iz dveh oblačkov. V levem je gumb, ki vodi nazaj na domačo stran, kolesce v desnem pa vodi na nastavitve (ta del zaenkrat še ni izdelan v celoti). Uporabnik bi mogoče rad imel drugačno abecedo, kot smo določili z lokalizacijo, kar bo lahko izbiral v nastavitvah.

Za enostavnejšo vizualizacijo seznama map v dveh stolpcih smo uporabili sistem mreže (*grid*) iz ogrodja *Bootstrap*. Za lepše animiranje seznama smo dodali knjižnico *iScroll*, ki jo priporočajo sploh za iOS hibridne aplikacije, saj prinese domoroden občutek. Imena pisav najprej pridobimo iz podatkovne baze, nato pa mape z imeni dodamo v dokument strani. Za tem se preberejo še tri črke za vsako pisavo, ki jih pridobimo urejene po združenem indeksu in dodamo na zaslon.

5.3 Stran za ustvarjanje pisave

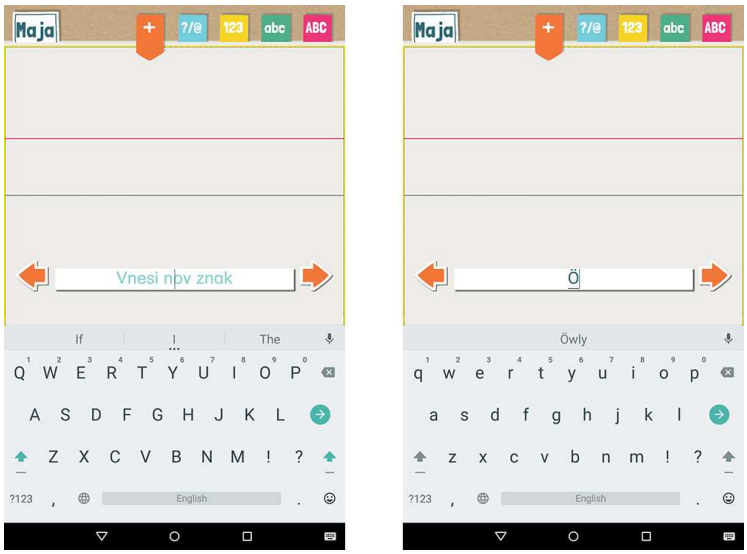
Na to stran lahko pridemo z domače strani ali s strani s shranjenimi pisavami. Videz je prikazan na sliki 5.4.



Slika 5.4: Ustvarjanje dveh malih tiskanih črk – pri y smo risali tudi ob meji canvasa, da se vidi prekrivanje elementov.

V zgornjem levem kotu je prikazano ime pisave. Desno od imena so barvni gumbi, s katerimi izberemo vrsto znakov, imenovali jih bomo zaznamki. Razvrščeni so po pomembnosti za otroka od desne proti levi: velike tiskane črke, male tiskane črke, številke, ločila, novi znak. Vrstni red črk oziroma abeceda in same črke so zaenkrat določeni pri prvem zagonu aplikacije (slovenščina ali angleščina). Možno je narediti vse številke. Ločila so določena kot tabela:

`["?", "/", "@", "!", ".", ",", "+", "-", "&", "\"", "#", "€", "$", "%", "(", ")", "*", ]`. Zaznamek novi znak omogoča dodajanje drugih unikod črk ali ločil oziroma hitro iskanje, če znak že obstaja, kot je prikazano na sliki 5.5.



Slika 5.5: Vstavljanje novega znaka.

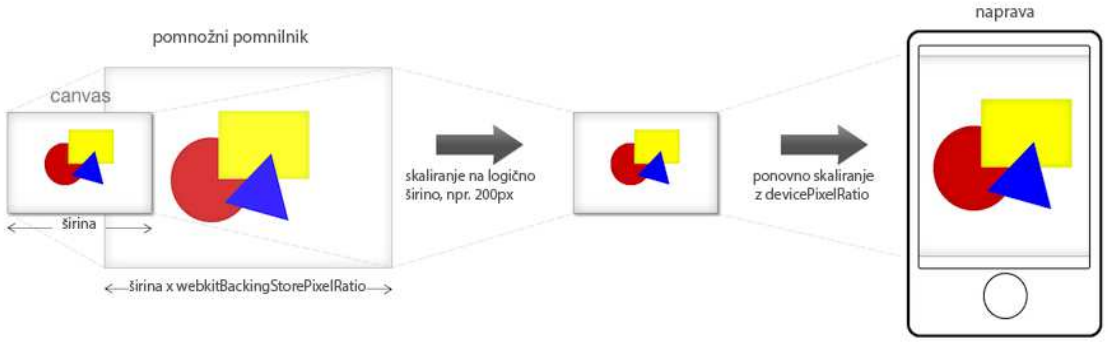
Uporabnik riše po osrednjem delu – canvas je na sliki 5.4 označen z rumenim okvirjem (v končni verziji bo odstranjen). Z vsakim dotikom spremljamo dogodek *onCanvasMouseMove* in ob premiku kličemo *updateCanvas* in rišemo črte. Vsako potezo shranimo kot niz koordinat x in y, v tabelo stilov dodamo debelino poteze, v tabelo točke stilov dolžino niza in povečamo števec potez. V osrednjem delu je tudi prikazan znak, ki ga ustvarjamo, in črte karakterja črk. Otrok lahko riše po velikem znaku, ali ga naredi drugače. S puščicami ciklično spreminjamo znak v izbrani skupini zaznamka. Ko spremenimo vrsto, je indeks trenutnega znaka shranjen v globalni spremenljivki, in sicer za vsak zaznamek posebej. Vsi indeksi so ponastavljeni, ko izberemo drugo pisavo. Narisan znak oziroma vse tabele shranimo v podatkovno bazo in dodamo združeni indeks abecede – to je lokacija v tabeli, kjer imamo združeno angleško in slovensko abecedo.

V spodnjem meniju imamo naslednje akcije (od leve proti desni): hiša vodi na domačo stran, črka A omogoča skrivanje in odkrivanje velikega znaka, z drsnikom (angl. slider) lahko spremenimo debeline črte risanja, sledi gumb za razveljavitev zadnje poteze in gumb za ponovni izris (angl. redo); puščica za naprej vodi naprej na stran za ustvarjanje voščilnic.

5.3.1 Zamegljeni canvas

Pri testiranju na visoko resolucijskem (hdpi) zaslonu smo se srečali z zamegljenim *canvasom* – narisana poteza ni bila jasna. Nekateri viri [13] navajajo, da je treba začetne in končne točke

premikati za 0.5, vendar ta popravek ni deloval. Ugotovili smo, da karkoli rišemo na canvas, se shranjuje v njegov pomožni pomnilnik (angl. backing store). Canvas ima tudi lastnost *webkitBackingStorePixelRatio*, ki pojasnjuje, kakšno je razmerje vrednosti v pomožnem pomnilniku v relaciji z elementom canvas. Brskalnik pri risanju canvasa na zaslon uporablja podatke iz pomožnega pomnilnika (slika 5.6) [15].



Slika 5.6: Shranjevanje canvasa in prikaz [15].

Visokokakovostni zasloni ustvarjajo bolj čiste in jasne slike tako, da več pikslov naprave stlačijo v manjši prostor. Zato en piksel naprave ni več enak enemu pikslu CSS. Skupaj s pojavom visokoresolucijskih zaslonov je nastala tudi lastnost razmerja pikslov na napravi (*devicePixelRatio*).

Ni nujno, da se vrednosti *devicePixelRatio* in *webkitBackingStorePixelRatio* ujemata. To pomeni, da samo s poznanim *devicePixelRatio* ne moremo vedeti, kako bo brskalnik sliko v canvasu skaliral. Če imamo canvas širine 200 pikslov, *webkitBackingStorePixelRatio* in *devicePixelRatio* enako 2, je naša shramba širine 400 pikslov. Pri risanju brskalnik pomanjša canvas glede na logično vrednost pikslov, ki je 200 pikslov. Potem sledi skaliranje z *devicePixelRatio*, kar pomeni, da je končna širina enaka 400 pikslov [15].

Če sta obe razmerji enaki, so narisane vrednosti pri shranjevanju avtomatsko povečane. Pri branju iz shrambe so najprej pomanjšane na logično vrednost in po skaliranju z *devicePixelRatio* spet enake narisanim. V našem primeru – razvijali smo na napravi Nexus 7 z Androidom 5.1.1 – smo imeli *webkitBackingStorePixelRatio* neznan in *devicePixelRatio* enak 2. Shranjene vrednosti so enake kot pri risanju, zato po skaliranju z razmerjem pikslov naprave dobimo skalirano in zamegljeno sliko (potezo).

Težavo smo rešili tako, da canvas skaliramo ročno. Velikost (višina in širina) canvasa izračunamo tako, da logično vrednost (v CSS določeno veličino) pomnožimo z razmerjem

devicePixelRatio/webkitBackingStorePixelRatio (če je *webkitBackingStorePixelRatio* neznan, vzamemo zanj vrednost 1 – enako velja za *devicePixelRatio*). Z enakim razmerjem skaliramo tudi obe koordinati vsake točke pri risanju in debelino črte. V podatkovno bazo shranjujemo neskalirane vrednosti.

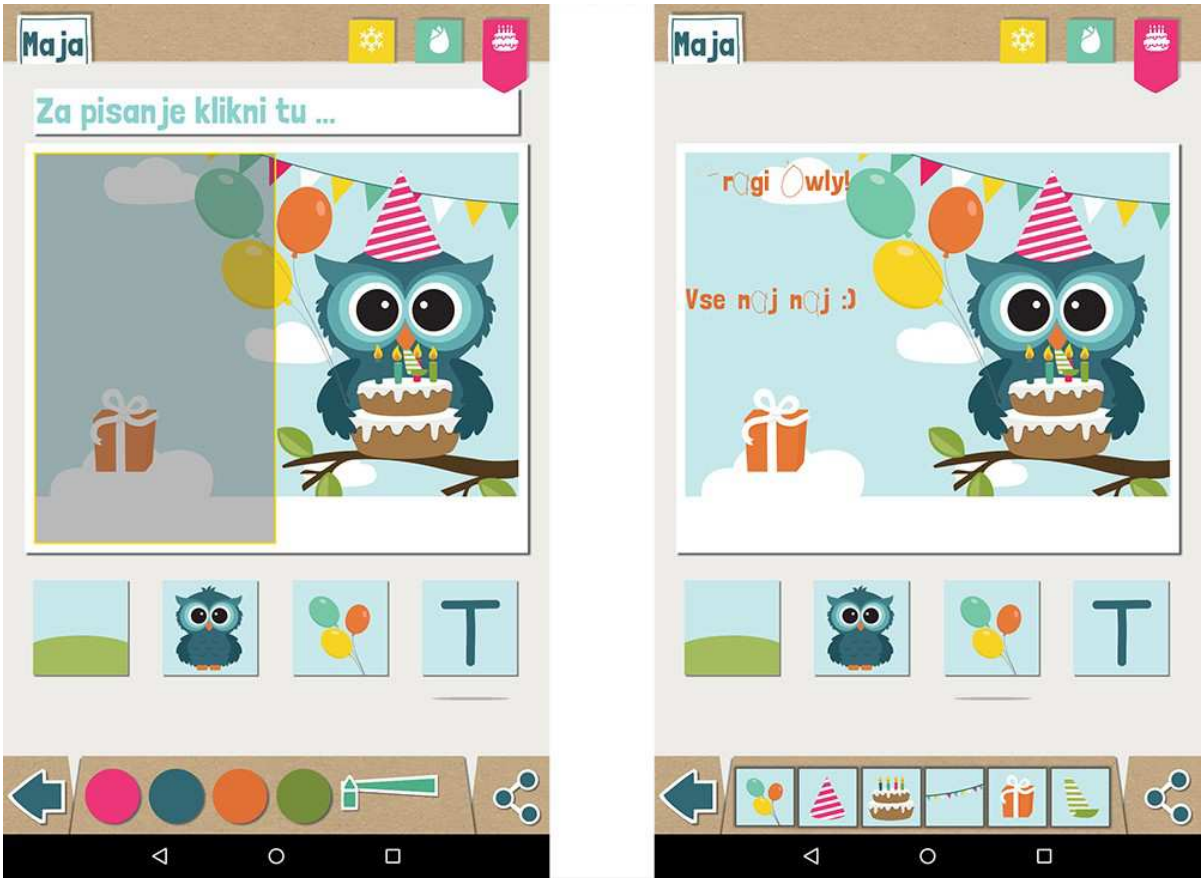
5.4 Stran za ustvarjanje voščilnic

Uporabnik lahko med ustvarjanjem pisave kadarkoli nadaljuje na stran za ustvarjanje voščilnic, tudi če ni naredil nobene črke. Na sliki 5.7 levo je prikazana stran s čisto osnovno voščilnico v osrednjem delu.

Slika 5.7: Levo je prikazana možnost izbire ozadja, desno so izbrani rožnata sova in dva dodatka.

Na levi strani zgoraj je ime pisave. Na desni strani so zaznamki, ki označujejo vrsto voščilnice oziroma za kakšno priložnost je pripravljena (od desne proti levi): za rojstni dan, za materinski dan in za novo leto – zadnji dve imata grafiki še v delu. Pod voščilnico je glavni meni, s katerim menjujemo možnost ustvarjanja, in podmeni v spodnji vrstici. Glavni meni

vsebuje gumb za izbiro ozadja, gumb za izbiro barve sove, gumb za izbiro dodatkov in gumb za ustvarjanje besedila. Na sliki 5.7 levo so v podmeniju prikazana vsa štiri možna ozadja; kjer je sova na desni strani, se elementi voščilnice zrcalijo preko osi y. Voščilnica je zapisana v oznaki svg (inline-svg), vsebovanim elementom smo ustrezno določili razrede CSS. Pri zrcaljenju vsem elementom dodamo atribut *transform*. Na sliki 5.7 desno je prikazana izbrana rožnata sova. Glavni lik je torej določen s štirimi razredi, vsakega barvamo z ustrezno roza ali modro različico. Možnih dodatkov je šest, uporabnik lahko izbere poljubno kombinacijo (vsak dodatek je svoj razred CSS). Pri besedilu lahko uporabnik spreminja barvo prek okroglih gumbov z določeno barvo in velikost besedila z drsnikom. Na voljo so štiri barve besedila: rožnata, oranžna, zelena in modra. Pisanje je prikazano na sliki 5.8. Če neke črke nismo ustvarili, je nadomeščena s črko iz pisave aplikacije.

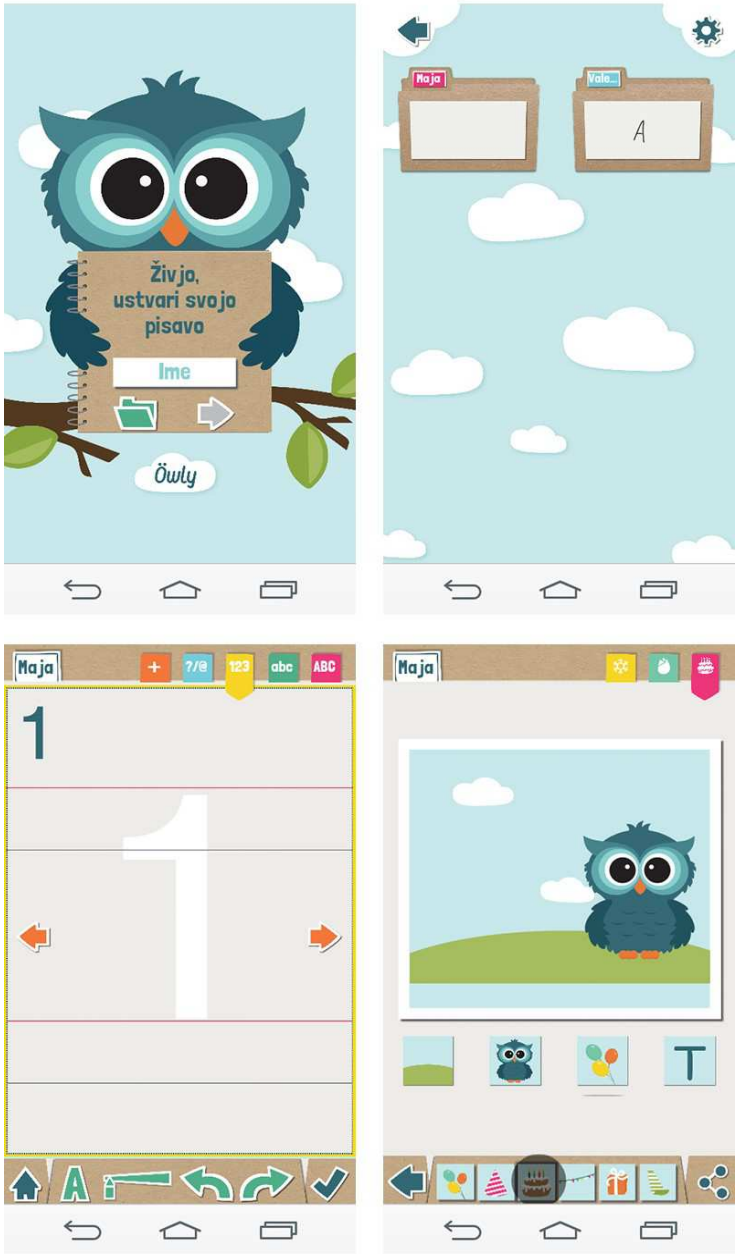


Slika 5.8: Levo je začetni zaslon pisanja z vsemi dodatki, desno končna voščilnica.

V spodnjem meniju imamo tudi skrajno levo puščico, ki nas vodi nazaj na pisanje (ustvarjena trenutna voščilnica se shrani), ter skrajno desno gumb za delitev voščilnice. Odpre se nam domorodno okno za deljenje.

5.5 Odzivna grafika

Čeprav je aplikacija izdelana upoštevajoč smernice za razvoj odzivnih spletnih aplikacij, je pripravljena grafika namenjena predvsem tablicam, za mobilne telefone pa je manj primerna, ker so elementi uporabniškega vmesnika na njih premajhni. Na sliki 5.9 so prikazani vsi štirje zasloni aplikacije. Na mapah je besedilo velikosti le 12 pikslov (3 mm na zaslonu). Na strani za ustvarjanje pisave je tudi z majhnimi otroškimi prsti težko zadeti želeni zaznamek, na strani za ustvarjanje voščilnic pa dodatke.



Slika 5.9: Vse strani na mobilni napravi, črna prosojna pika označuje prst.

Poglavje 6 Sklepne ugotovitve

Aplikacijo Öwly smo izdelali v sodelovanju s študentko Naravoslovnotehniške fakultete, ki je v okviru svojega diplomskega dela izdelala zasnovo mobilne aplikacije in ustvarila grafiko. Želja je, da bi aplikacijo objavili vsaj v trgovini Google Play, ko bo vsa grafika dokončana (implementacija je pripravljena tako, da jo bo možno hitro vključiti), torej z vsemi tremi priložnostmi pri ustvarjanju voščilnic, mogoče pa tudi z nastavitvami.

Otroci imajo zelo radi zvočne učinke, zato bi ti spadali med pomembnejše dodatke. Pri ustvarjanju voščilnic bi lahko omogočili večjo izbiro barv tako za besedilo kot za sovico ter ponudili celo prosto premikanje elementov. Pri izdelavi voščilnic bi lahko vključili tudi dodajanje osnovnih likov ali celo manjših slik. Pri izbrani pisavi za ozadje pisanja bi bilo dobro nadomestiti zaprto številko 4 z odprto. Ker prikazujemo črke kot slike SVG, imamo možnost, da bi ustvarjeno pisavo izvozili ter uporabili tudi izven aplikacije (treba je poudariti, da se otrok ne drži nobenih načel pri risanju, zato je vprašljivo, koliko bi bila ta uporabna). S tehničnega vidika bi lahko zamenjali knjižnico *jQuery.js* z manjšo in bolj za mobilne naprave namenjeno *Zepto.js*.

Cordova s HTML5 elementoma canvas in svg ustreza ustvarjanju, prikazovanju in spreminjanju lastnosti črk. Manjša pomanjkljivost je počasnejše branje iz podatkovne baze, kot če bi delali samo v domorodni kodi – problem je pri sprejemanju veliko podatkov hkrati iz domorodne kode v Javascript. Dobra stran je, da bi lahko v kratkem času naredili oziroma dogradili aplikacijo tudi za drugo platformo. Za iOS bi za testiranje potrebovali iPad in račun razvijalca (tudi računalnik z operacijskim sistemom Mac OS). Možno je, da bi aplikacija delovala počasi ter bi morali dodati vtičnik za uporabo pogona *wkwebview*. Za Windows Phone in BlackBerry bi morali najprej preveriti podporo vseh vtičnikov, ki smo jih dodali.

Spoznavanje s Cordovo je počasen proces, prav tako tudi priprava in naložitev razvojnega okolja. Po uspešni namestitvi pa Cordova nekomu, ki ima zelo dobro znanje spletnih tehnologij, predvsem pa Javascripta, zagotovo omogoča hitro delo. Če razvijamo za več mobilnih operacijskih sistemov, moramo upoštevati več posebnosti in potrebnega je več testiranja. Z večanjem moči naprave raste tudi zmožnost aplikacije v spletnem pogledu.

Trenutno je pomembno zavedanje in upoštevanje ciljne skupine, tako da je veliko knjižnic Javascript namenjenih samo mobilnim napravam, ki so manjše in imajo pogosto dodane dogodke, povezane z dotikom na zaslon. Pojavlja se vprašanje, ali obstaja možnost, da bi število hibridnih mobilnih aplikacij kdaj prekašalo domorodne.

Literatura

[1] Vir za razvijalce Android aplikacij. *Android Developers* [Online]. Dosegljivo: https://developer.android.com/index.html.

[2] Statistika prenosov iOS aplikacij. *Apple store downloads 2015* [Online]. Dosegljivo: http://www.statista.com/statistics/263794/number-of-downloads-from-the-apple-app-store.

[3] Tipografija [Online]. Dosegljivo: https://eucbeniki.sio.si/test/lum9/2476.

[4] J. S. Maria. On Web Typography. A book apart. 2014.

[5] Razlika med pisali za zaslon. *Not All Tablet Styluses Are Equal: Capacitive, Wacom, and Bluetooth Explained* [Online]. Dosegljivo: http://www.howtogeek.com/177376/not-all-tablet-styluses-are-equal-capacitive-wacom-and-bluetooth-explained.

[6] Aktivno pisalo za zaslon s povezovanjem preko bluetootha. *Intuos creative stylus 2 getting started*. [Online]. Dosegljivo: http://www.wacom.com/en-us/getting-started/intuos-creative-stylus-2-getting-started.

[7] Pravila in napotki za objavo aplikacije na App Store (2015). *App Store Review Guidelines* [Online]. Dosegljivo: https://developer.apple.com/app-store/review/guidelines.

[8] Hibridne in domorodne aplikacije. (oktober 2014). *Hybrid mobile apps* [Online]. Dosegljivo: http://www.smashingmagazine.com/2014/10/providing-a-native-experience-with-web-technologies.

[9] J. M. Wargo. Apache Cordova 3 Programming. Addison-Wesley, 2013.

[10] T. Myer. Beginning Phonegap. John Wiley & Sons, Inc, 2012.

[11] Dokumentacija Cordove. *Apache Cordova API Documentation* [Online]. Dosegljivo: http://cordova.apache.org/docs/en/edge.

[12] Vodiči za programerje. *Tutorials for ...* [Online]. Dosegljivo: http://www.tutorialspoint.com.

[13] Element canvas. *Canvas – Dive Into HTML5* [Online]. Dosegljivo: http://diveintohtml5.info/canvas.html.

[14] Podpora za HTML5 in CSS3. *Can I use... Support tables for HTML5, CSS3, etc.* [Online]. Dosegljivo: http://caniuse.com.

[15] Canvas višje resolucije. *High DPI Canvas* [Online]. Dosegljivo: http://www.html5rocks.com/en/tutorials/canvas/hidpi/.